HW01WP_ML_final

June 27, 2023

1 HW 1 Analysis Problems

Start by importing necessary packages

[ ]:
```
# import packages
import pandas as pd
import numpy as np
import seaborn as sns
```

1.1 1. Read an Excel data table that you create

a. List each of the variable names in the data table and classify each as continuous or categorical.

Gender = categorical

Weight = continuous

Height = continuous

b. Create a folder (directory) named ***data*** *that exists adjacent to your HW 1 Jupyter Notebook in your computer's file system. Then enter the above data into an Excel worksheet and save as file DataEg.xlsx into this data folder. (Or create the data folder on Google Drive if using Colab.)*

Read the data directly from the Excel file you created on your computer system into a Python data table.

[ ]:
```
# Import file into data frame with read_excel
df = pd.read_excel('/Users/chasecarlson/Documents/GSCM Course Materials/GSCM␣
↪575 Machine Learning in Business/Python Pjojects/GSCM-575-ML/data/DataEg.
↪xlsx')
df.head()
```

[ ]:

	Gender	Weight	Height
0	F	150	66.0
1	F	138	66.0
2	M	240	NaN
3	M	178	71.0
4	F	130	64.0

c. Display the entire data table.

[ ]:
```
# View the data frame by calling the alias
df
# print(df)
```

[ ]:
```
  Gender  Weight  Height
0      F     150    66.0
1      F     138    66.0
2      M     240     NaN
3      M     178    71.0
4      F     130    64.0
5      M     200    74.0
6      F     140    70.0
7      M     220    77.0
```

d. Verify that the data values for the variables in the data table are stored within the analysis system in the intended format, that is, character string, integer numeric, or float numeric.

[ ]:
```
# Use dtypes to view data types
df.dtypes
```

[ ]:
```
Gender     object
Weight      int64
Height    float64
dtype: object
```

[ ]:
```
# Or use .info() to view data types + additional info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  8 non-null      object
 1   Weight  8 non-null      int64
 2   Height  7 non-null      float64
dtypes: float64(1), int64(1), object(1)
memory usage: 320.0+ bytes
```

e. Compare the data stored in Excel and then compare to the representation of the data stored in Python. What does NaN refer to in the Python data table?

NaN refers to blank cells, or missing data. NaN stands for “Not a Number”

1.2 2. Read a large data set from the web

See the codebook for the definition of the variables.

Data: http://web.pdx.edu/~gerbing/data/bank-full.csv

Codebook: http://web.pdx.edu/~gerbing/data/bank-full.xlsx *a. Read the data into Python.*

```
[ ]: # Import data from web using read_csv
     df2 = pd.read_csv('https://web.pdx.edu/~gerbing/data/bank-full.csv')
     df2.head()
```

```
[ ]:    age           job  marital  education default  balance housing loan  \
     0   58    management  married   tertiary      no     2143     yes   no
     1   44    technician   single  secondary      no       29     yes   no
     2   33  entrepreneur  married  secondary      no        2     yes  yes
     3   47   blue-collar  married    unknown      no     1506     yes   no
     4   33       unknown   single    unknown      no        1      no   no

        contact  day month  duration  campaign  pdays  previous poutcome   y
     0  unknown    5   may       261         1     -1         0  unknown  no
     1  unknown    5   may       151         1     -1         0  unknown  no
     2  unknown    5   may        76         1     -1         0  unknown  no
     3  unknown    5   may        92         1     -1         0  unknown  no
     4  unknown    5   may       198         1     -1         0  unknown  no
```

b. How many rows of data? Columns of data?

```
[ ]: # View rows and columns using .shape
     df2.shape
     # 45,211 rows & 17 columns
```

```
[ ]: (45211, 17)
```

c. List the first ten rows of data and the variable names.

```
[ ]: # List first 10 rows using .head()
     df2.head(10)
```

```
[ ]:    age           job   marital  education default  balance housing loan  \
     0   58    management   married   tertiary      no     2143     yes   no
     1   44    technician    single  secondary      no       29     yes   no
     2   33  entrepreneur   married  secondary      no        2     yes  yes
     3   47   blue-collar   married    unknown      no     1506     yes   no
     4   33       unknown    single    unknown      no        1      no   no
     5   35    management   married   tertiary      no      231     yes   no
     6   28    management    single   tertiary      no      447     yes  yes
     7   42  entrepreneur  divorced   tertiary     yes        2     yes   no
     8   58       retired   married    primary      no      121     yes   no
     9   43    technician    single  secondary      no      593     yes   no

        contact  day month  duration  campaign  pdays  previous poutcome   y
     0  unknown    5   may       261         1     -1         0  unknown  no
     1  unknown    5   may       151         1     -1         0  unknown  no
     2  unknown    5   may        76         1     -1         0  unknown  no
```

```
3  unknown    5   may        92          1      -1          0  unknown  no
4  unknown    5   may       198          1      -1          0  unknown  no
5  unknown    5   may       139          1      -1          0  unknown  no
6  unknown    5   may       217          1      -1          0  unknown  no
7  unknown    5   may       380          1      -1          0  unknown  no
8  unknown    5   may        50          1      -1          0  unknown  no
9  unknown    5   may        55          1      -1          0  unknown  no
```

d. Use Python to identify and distinguish some numeric variables from character string variables.

[ ]:
```
# Check the data types for each variable.
df2.dtypes
```

[ ]:
```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

e. All variables with non-numeric characters as data values are necessarily categorical variables. Identify any categorical numerical variables.

[ ]:
```
# Identifying categorical numerical variables.
# Start by viewing the data set to interpret what the values represent.
print(df2)
```

```
       age           job   marital  education default  balance housing loan  \
0       58    management   married   tertiary      no     2143     yes   no
1       44    technician    single  secondary      no       29     yes   no
2       33  entrepreneur   married  secondary      no        2     yes  yes
3       47   blue-collar   married    unknown      no     1506     yes   no
4       33       unknown    single    unknown      no        1      no   no
…     …            …       …          …         …       …      …   …
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
```

```
45209   57    blue-collar   married  secondary      no      668       no   no
45210   37   entrepreneur   married  secondary      no     2971       no   no

         contact  day month  duration  campaign  pdays  previous poutcome    y
0        unknown    5   may       261         1     -1         0  unknown   no
1        unknown    5   may       151         1     -1         0  unknown   no
2        unknown    5   may        76         1     -1         0  unknown   no
3        unknown    5   may        92         1     -1         0  unknown   no
4        unknown    5   may       198         1     -1         0  unknown   no
…            …   …   …        …        …      …        …      …   …
45206   cellular   17   nov       977         3     -1         0  unknown  yes
45207   cellular   17   nov       456         2     -1         0  unknown  yes
45208   cellular   17   nov      1127         5    184         3  success  yes
45209  telephone   17   nov       508         4     -1         0  unknown   no
45210   cellular   17   nov       361         2    188        11    other   no

[45211 rows x 17 columns]
```

Categorical numerical variables could include: >

day - The day of the month can be treated as a categorical variable.

age - Age could be treated as a categorical variable if specific age brackets are needed to analyze the campaign (i.e. “18-15”, “26-35”, “36-45”, etc…)

campaign - If the numbers correspond to a campaign number it would not hold quantitative value. However, I would need more context about the data set to determine what the values represent.

f. List the frequencies of the different types of recorded levels of marital status.

[ ]:
```python
# Listing the count each marital status appears in the data set using␣
↪value_counts()
df2.marital.value_counts()
```

[ ]:
```
married     27214
single      12790
divorced     5207
Name: marital, dtype: int64
```

g. Define the variable marital *as a categorical variable with values listed in the order of ‘single’, ‘married’, and ‘divorced’.*

[ ]:
```python
# Defining 'marital' column as categorical variable in the order of 'single',␣
↪'married', and 'divorced'
df2['marital'] = pd.Categorical(df2['marital'], categories=['single',␣
↪'married', 'divorced'], ordered=True)
df2.dtypes
```

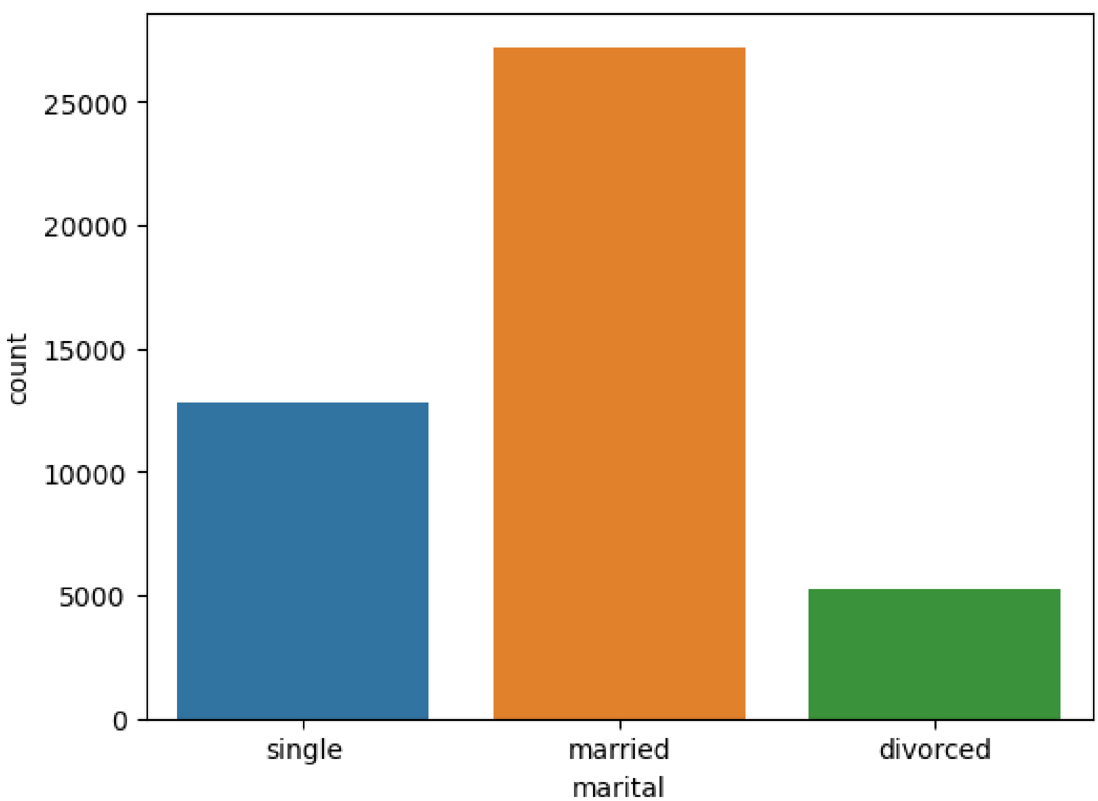
[ ]:
```
age              int64
job             object
```

```
marital      category
education      object
default        object
balance         int64
housing        object
loan           object
contact        object
day             int64
month          object
duration        int64
campaign        int64
pdays           int64
previous        int64
poutcome       object
y              object
dtype: object
```

h. Create the bar chart of marital.

[ ]:
```python
# Creating a bar plot for marital status using seaborn
sns.countplot(df2, x='marital')
```

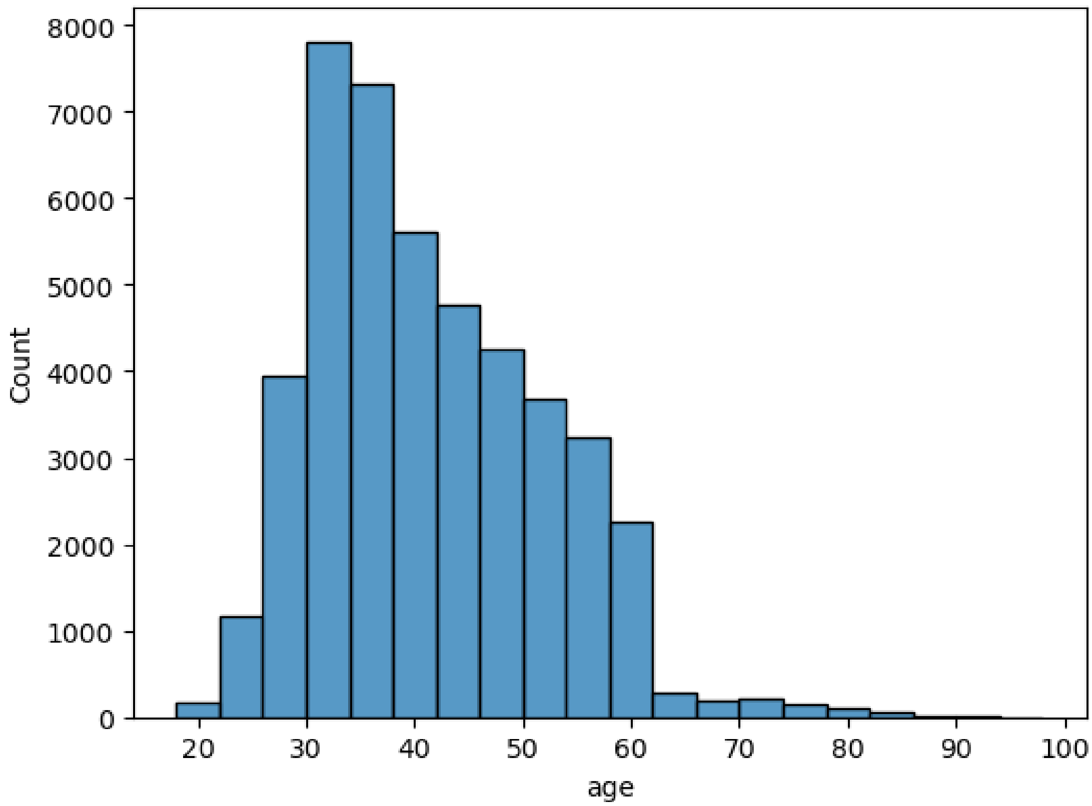
[ ]:
```
<Axes: xlabel='marital', ylabel='count'>
```



i. Run the histogram of age with a bin width of 4. What is the advantage of this histogram over the default version?

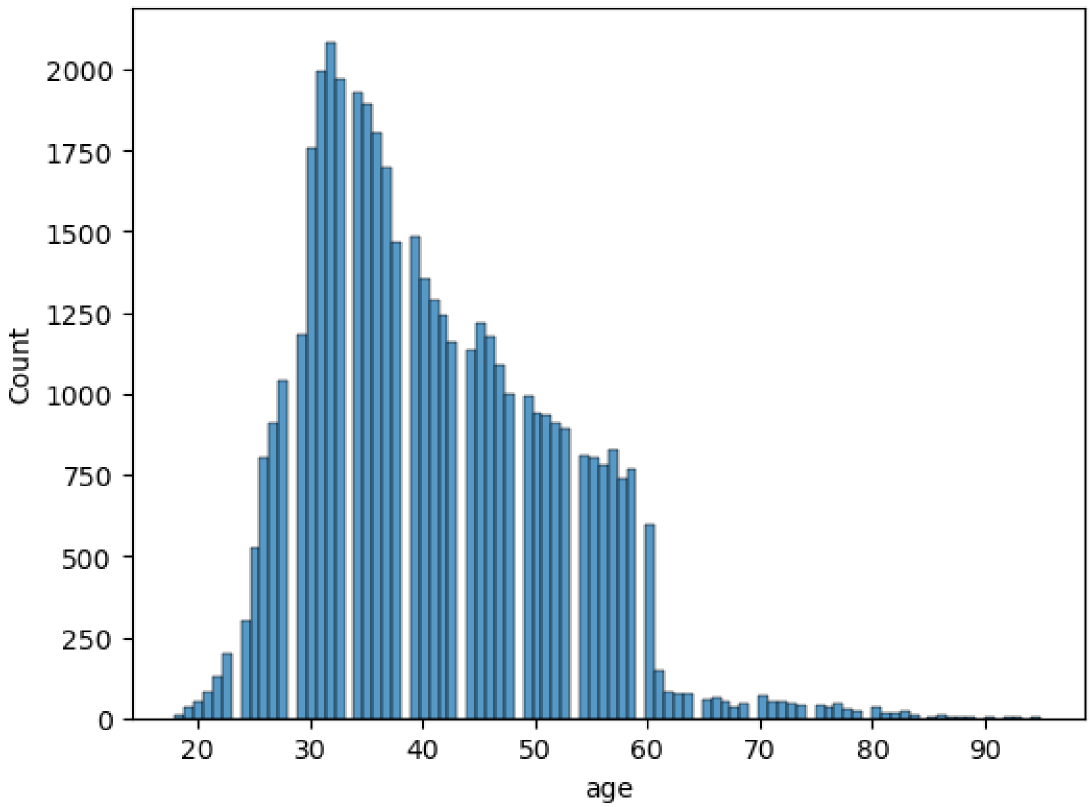
[ ]:
```
# Creating histogram of age with bin width of 4 (years).
sns.histplot(df2, x='age', binwidth=4)
```

[ ]: <Axes: xlabel='age', ylabel='Count'>



[ ]:
```
# View default histogram of age column
sns.histplot(df2, x='age')
```

[ ]: <Axes: xlabel='age', ylabel='Count'>



Using a binwidth of 4 creates a better representation of the data than the default histogram, and it helps produce more meaningful results. The default version is too specific (1 yr bins) and has gaps in the plot.